**ỨNG DỤNG NETWORK SCANNER**   
**WITH PYTHON**

\*Lò Minh Thuận

# I. BASIC NETWORK SCANER

Đoạn code trên thực hiện một quá trình quét cổng (port scanning) để kiểm tra xem các cổng trên một địa chỉ IP hoặc tên miền nhất định có mở hay không.

Dưới đây là giải thích chi tiết về từng dòng code:

#ver1.0

import socket

import subprocess

from datetime import datetime

subprocess.call('clear', shell=True)

target = input("Moi nhap dia chi IP va ten mien: ")

def port\_scan(target):

    try:

        ip = socket.gethostbyname(target)

        print("-" \* 50)

        print("Scanning target:", ip)

        print("Time started:", datetime.now())

        print("-" \* 50)

        for port in range(1, 65635):

          sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

           sock.settimeout(1)# Đặt thời gian timeout cho việc kết nối socket tới 1 giây.

            result = sock.connect\_ex((ip, port))

            if result == 0:

                print("Port {}: Open".format(port))

            sock.close()

    except socket.gaierror:

        print("Hostname could not be resolved.")

    except socket.error:

        print("Could not connect to the server.")

port\_scan(target)

Đoạn mã này thực hiện một quá trình quét cổng đơn giản để kiểm tra xem các cổng trên một địa chỉ IP hoặc tên miền nhất định.

*Ưu điểm:*

+ Dễ dàng sử dụng, tùy chỉnh

+ Tiện lợi, kiểm tra đơn giản

+ Câu lệnh python thuần túy, sử dụng các thư viện socket, subprocess, datetime có trong python

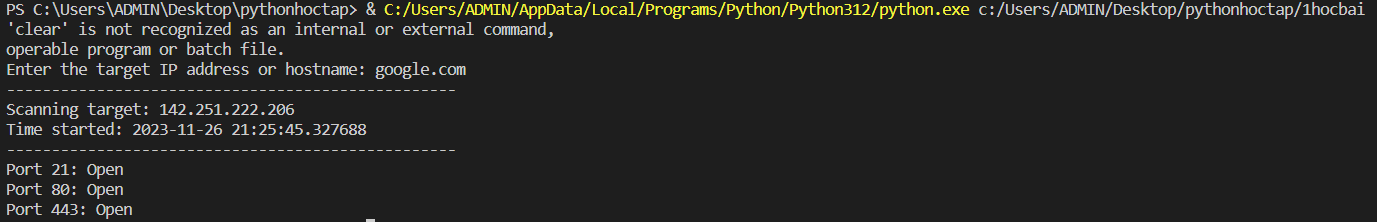
*Nhược điểm:*

+ Thời gian chạy quét dựa vào thự viện, tốn nhiều thời gian

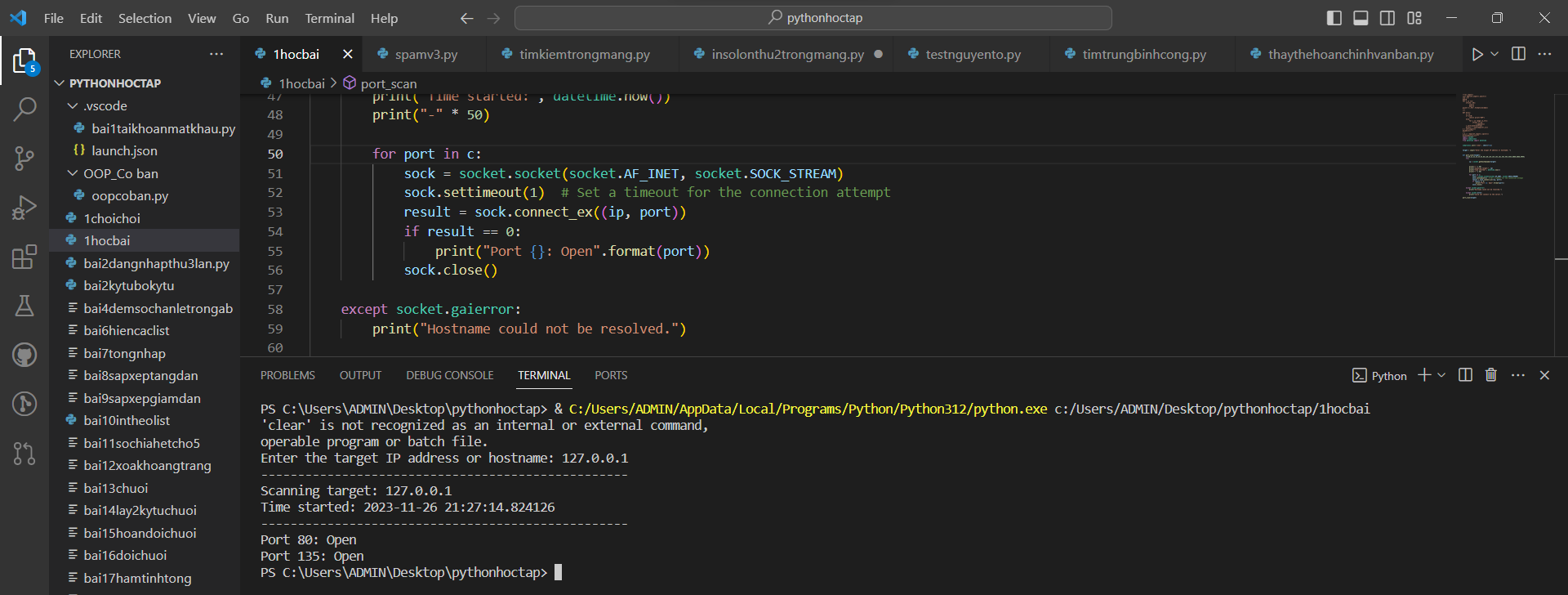
+ Không chỉnh sửa như ip, quét tcp hay udp

+ Nếu không tự import cổng thì sẽ chạy từ 1 - > 65635, mất nhiều thời gian, tài nguyên.

*Kết quả:*



*Hình 1.1: Kết quả quét hostname: google.com*



*Hình 1.2: Kết quả quét IP 127.0.0.1*

# II. ADVANCE NETWORK SCANER

#!/bin/python3/ver2.0

def atoi(text):

    return int(text) if text.isdigit() else text

def natural\_keys(text):

    return [ atoi(c) for c in re.split(r'(\d+)', text) ]

def getServiceName(port, proto):

    try:

        name = socket.getservbyport(int(port), proto)

    except:

        return None

    return name

def kill\_process():

    print(f"\n{bcolors.RED}Closing process....{bcolors.ENDC}")

    if hasattr(signal, 'SIGKILL'):

        kill(pid, signal.SIGKILL)

    else:

        kill(pid, signal.SIGABRT)

    exit()

def check\_ip(host,port,type\_port):

    try:

        if (type\_port == "TCP"):

            s = socket.socket()

            s.settimeout(4)

            s.connect((host, int(port)))

        elif (type\_port == "UDP"):

            s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM, socket.IPPROTO\_UDP)

            s.sendto("Khanh".encode('utf\_8'), (host, port))

            s.settimeout(2)

            s.recvfrom(1024)

    except TimeoutError:

        if (type\_port == "UDP"):

            serv = getServiceName(port, type\_port)

            if not serv:

                pass

            else:

                s.close()

                open.append(f"{port}    {serv}")

    except:

        pass

    else:

        s.close()

        try:

            temp=str(info\_port[port])

        except KeyError:

            temp="unknown"

        open.append(f"{port}    {temp}")

def main(host,min,max,timeout,type\_port):

    print(f"\n{bcolors.GREEN}Scanning {type\_port}... ({bcolors.BLUE}{host}{bcolors.GREEN}) [{bcolors.CYAN}{min}{bcolors.GREEN}-{bcolors.CYAN}{max}{bcolors.GREEN}]{bcolors.ENDC}")

    for port in range(min,max+1,1):

        try:

            Thread(target=check\_ip, args=(host,port,type\_port)).start()

            i=int((100/(max-min))\*(port-min))

            stdout.write(f"\r[%-25s] %d%% ({bcolors.YELLOW}{port}{bcolors.ENDC})" % ('='\*int(i/4), i))

            stdout.flush()

            sleep(timeout)

        except RuntimeError as e:

            print(f"{bcolors.RED}\n\nERROR: {e} at {port}")

            print(f"{bcolors.YELLOW}You can try min and max port:\n Min: 1, Max: {port-6}\n Min: 1000, Max: {port-6+1000}\n Min: 3000, Max: {port-6+3000}{bcolors.ENDC}")

            kill\_process()

        except KeyboardInterrupt:

            break

    print(f"\n{bcolors.GREEN}Waiting for the process complete (10s)...{bcolors.ENDC}")

    sleep(10)

    open.sort(key=natural\_keys)

    if (open == []):

        print(f"\n{bcolors.RED}There are no port {type\_port} open from ({bcolors.BLUE}{host}{bcolors.RED})")

    else:

        print(f"\n{bcolors.CYAN}Port {type\_port} open from ({bcolors.BLUE}{host}{bcolors.CYAN}):{bcolors.YELLOW}")

        for i in open:

            temp=i.split()

            print(f"{bcolors.YELLOW}{temp[0]}/{type\_port.lower()}   {bcolors.BLUE}{temp[1]}{bcolors.ENDC}")

    print(bcolors.ENDC,end="")

    kill\_process()

if (\_\_name\_\_ == "\_\_main\_\_"):

    from time import sleep

    from threading import Thread

    from os import kill, getpid

    from sys import stdout

    import socket, signal, re

    while True:

        print(f"Use color? [Y/n]: ", end="")

        temp=str(input())

        if (temp == "Y") or (temp == "y"):

            class bcolors:

                BLUE = '\033[94m'

                CYAN = '\033[96m'

                GREEN = '\033[92m'

                YELLOW = '\033[93m'

                RED = '\033[91m'

                ENDC = '\033[0m'

            break

        elif (temp == "N") or (temp == "n"):

            class bcolors:

                BLUE = ''

                CYAN = ''

                GREEN = ''

                YELLOW = ''

                RED = ''

                ENDC = ''

            break

    info\_port={1:"tcpmux", 3:"compressnet", 7:"echo", 9:"discard", 13:"daytime", 17:"qotd", 19:"chargen", 20:"ftp-data", 21:"ftp", 22:"ssh", 23:"telnet", 25:"smtp", 53:"domain", 80:"http", 106:"pop3pw", 110:"pop3", 111:"rpcbind", 113:"ident", 135:"msrpc", 139:"netbios-ssn", 143:"imap", 199:"smux", 256:"fw1-secureremote", 416:"silverplatter", 443:"https", 445:"microsoft-ds", 500:"isakmp", 515:"printer", 554:"rtsp", 587:"submission", 901:"samba-swat", 902:"iss-realsecure", 993:"imaps", 995:"pop3s", 1025:"NFS-or-IIS", 1063:"kyoceranetdev", 1068:"instl\_bootc", 1131:"caspssl", 1720:"h323q931", 1723:"pptp", 2007:"dectalk", 2111:"kx", 2121:"ccproxy-ftp", 2222:"EtherNetIP-1", 2382:"ms-olap3", 2500:"rtsserv", 2608:"wag-service", 2800:"acc-raid", 3005:"deslogin", 3268:"globalcatLDAP", 3306:"mysql", 3389:"ms-wbt-server", 3580:"nati-svrloc", 3689:"rendezvous", 3703:"adobeserver-3", 3814:"neto-dcs", 3826:"wormux", 3869:"ovsam-mgmt", 3871:"avocent-adsap", 3878:"fotogcad", 3889:"dandv-tester", 3920:"exasoftport1", 3945:"emcads", 3971:"lanrevserver", 3986:"mapper-ws\_ethd", 3995:"iss-mgmt-ssl", 3998:"dnx", 4000:"remoteanything", 4003:"pxc-splr-ft", 4004:"pxc-roid", 4005:"pxc-pin", 4006:"pxc-spvr", 4045:"lockd", 4126:"ddrepl", 4129:"nuauth", 4224:"xtell", 4242:"vrml-multi-use", 4279:"vrml-multi-use", 4444:"krb524", 4445:"upnotifyp", 4449:"privatewire", 4567:"tram", 4662:"edonkey", 4848:"appserv-http", 4899:"radmin", 4900:"hfcs", 4998:"maybe-veritas", 5000:"upnp", 5001:"commplex-link", 5002:"rfe", 5003:"filemaker", 5004:"avt-profile-1", 5009:"airport-admin", 5030:"surfpass", 5033:"jtnetd-server", 5050:"mmcc", 5051:"ida-agent", 5054:"rlm-admin", 5060:"sip", 5061:"sip-tls", 5080:"onscreen", 5087:"biotic", 5100:"admd", 5101:"admdog", 5102:"admeng", 5120:"barracuda-bbs", 5190:"aol", 5200:"targus-getdata", 5221:"3exmp", 5222:"xmpp-client", 5225:"hp-server", 5226:"hp-status", 5269:"xmpp-server", 5280:"xmpp-bosh", 5298:"presence", 5357:"wsdapi", 5405:"pcduo", 5414:"statusd", 5431:"park-agent", 5432:"postgresql", 5500:"hotline", 5510:"secureidprop", 5550:"sdadmind", 5555:"freeciv", 5560:"isqlplus", 5566:"westec-connect", 5631:"pcanywheredata", 5633:"beorl", 5666:"nrpe", 5678:"rrac", 5679:"activesync", 5718:"dpm", 5730:"unieng", 5800:"vnc-http", 5801:"vnc-http-1", 5802:"vnc-http-2", 5859:"wherehoo", 5900:"vnc", 5901:"vnc-1", 5902:"vnc-2", 5903:"vnc-3", 5910:"cm", 5911:"cpdlc", 5963:"indy", 5987:"wbem-rmi", 5988:"wbem-http", 5989:"wbem-https", 5998:"ncd-diag", 5999:"ncd-conf", 6000:"X11", 6001:"X11:1", 6002:"X11:2", 6003:"X11:3", 6004:"X11:4", 6005:"X11:5", 6006:"X11:6", 6007:"X11:7", 6009:"X11:9", 6025:"x11", 6059:"X11:59", 6100:"synchronet-db", 6101:"backupexec", 6106:"isdninfo", 6112:"dtspc", 6123:"backup-express", 6346:"gnutella", 6389:"clariion-evr01", 6502:"netop-rc", 6510:"mcer-port", 6543:"mythtv", 6547:"powerchuteplus", 6566:"sane-port", 6567:"esp", 6580:"parsec-master", 6666:"irc", 6667:"irc", 6668:"irc", 6669:"irc", 6689:"tsa", 6699:"napster", 6788:"smc-http", 6789:"ibm-db2-admin", 6881:"bittorrent-tracker", 6901:"jetstream", 6969:"acmsoda", 7000:"afs3-fileserver", 7001:"afs3-callback", 7002:"afs3-prserver", 7004:"afs3-kaserver", 7007:"afs3-bos", 7019:"doceri-ctl", 7025:"vmsvc-2", 7070:"realserver", 7100:"font-service", 7200:"fodms", 7201:"dlip", 7402:"rtps-dd-mt", 7443:"oracleas-https", 7627:"soap-http", 7676:"imqbrokerd", 7741:"scriptview", 7777:"cbt", 7778:"interwise", 7800:"asr", 7937:"nsrexecd", 7938:"lgtomapper", 7999:"irdmi2", 8000:"http-alt", 8001:"vcom-tunnel", 8002:"teradataordbms", 8007:"ajp12", 8008:"http", 8009:"ajp13", 8010:"xmpp", 8021:"ftp-proxy", 8022:"oa-system", 8042:"fs-agent", 8080:"http-proxy", 8081:"blackice-icecap", 8082:"blackice-alerts", 8083:"us-srv", 8086:"d-s-n", 8087:"simplifymedia", 8088:"radan-http", 8090:"opsmessaging", 8100:"xprint-server", 8122:"army2-server", 8181:"intermapper", 8192:"sophos", 8193:"sophos", 8194:"sophos", 8200:"trivnet1", 8292:"blp3", 8300:"tmi", 8333:"bitcoin", 8383:"m2mservices", 8400:"cvd", 8402:"abarsd", 8443:"https-alt", 8500:"fmtp", 8600:"asterix", 8800:"sunwebadmin", 8873:"dxspider", 8888:"sun-answerbook", 8899:"ospf-lite", 9000:"cslistener", 9001:"tor-orport", 9002:"dynamid", 9009:"pichat", 9010:"sdr", 9011:"d-star", 9040:"tor-trans", 9050:"tor-socks", 9080:"glrpc", 9081:"cisco-aqos", 9090:"zeus-admin", 9091:"xmltec-xmlmail", 9100:"jetdirect", 9101:"jetdirect", 9102:"jetdirect", 9103:"jetdirect", 9111:"DragonIDSConsole", 9200:"wap-wsp", 9207:"wap-vcal-s", 9418:"git", 9500:"ismserver", 9535:"man", 9593:"cba8", 9594:"msgsys", 9595:"pds", 9618:"condor", 9666:"zoomcp", 9876:"sd", 9878:"kca-service", 9898:"monkeycom", 9900:"iua", 9929:"nping-echo", 9998:"distinct32", 9999:"abyss", 10000:"snet-sensor-mgmt", 10001:"scp-config", 10002:"documentum", 10003:"documentum\_s", 10004:"emcrmirccd", 10009:"swdtp-sv", 10010:"rxapi", 10082:"amandaidx", 11110:"sgi-soap", 11111:"vce", 11967:"sysinfo-sp", 12000:"cce4x", 12345:"netbus", 13722:"netbackup", 13782:"netbackup", 13783:"netbackup", 14000:"scotty-ft", 14444:"nso-server", 14445:"nro-server", 15000:"hydap", 15002:"onep-tls", 15660:"bex-xr", 16000:"fmsas", 16001:"fmsascon", 16080:"osxwebadmin", 16992:"amt-soap-http", 16993:"amt-soap-https", 19283:"keysrvr", 19315:"keyshadow", 20000:"dnp", 20005:"btx", 20222:"ipulse-ics", 27000:"flexlm0", 30000:"ndmps", 31337:"Elite", 32768:"filenet-tms", 32769:"filenet-rpc", 32770:"sometimes-rpc3", 32771:"sometimes-rpc5", 32772:"sometimes-rpc7", 32773:"sometimes-rpc9", 32774:"sometimes-rpc11", 32775:"sometimes-rpc13", 32776:"sometimes-rpc15", 32777:"sometimes-rpc17", 32778:"sometimes-rpc19", 32779:"sometimes-rpc21", 32780:"sometimes-rpc23", 38292:"landesk-cba", 42510:"caerpc", 44442:"coldfusion-auth", 44443:"coldfusion-auth", 49400:"compaqdiag", 50000:"ibm-db2", 50002:"iiimsf", 62078:"iphone-sync"}

    global open, pid

    open=[]

    min=0

    max=0

    timeout=999

    pid = getpid()

    try:

        host=str(input(f"{bcolors.YELLOW}IP: {bcolors.GREEN}")).replace("https://","").replace("http://","").split("/")

    except KeyboardInterrupt:

        kill\_process()

    try:

        host=str(socket.gethostbyname(host[0]))

    except:

        host=str(host[0])

    while (min < 1) or (min > 65535):

        try:

            min=int(input(f"{bcolors.YELLOW}Min port: {bcolors.GREEN}"))

        except KeyboardInterrupt:

            kill\_process()

        except:

            print(f"\n{bcolors.CYAN} Integer number [1-65535]\n First port while scanning\n{bcolors.ENDC}")

            min=0

    while (max < min) or (max > 65535):

        try:

            max=int(input(f"{bcolors.YELLOW}Max port: {bcolors.GREEN}"))

        except KeyboardInterrupt:

            kill\_process()

        except:

            print(f"\n{bcolors.CYAN} Integer number [1-65535]\n Last port while scanning\n Large port can take quite a while\n{bcolors.ENDC}")

            max=0

    if (min == max):

        if (max == 65535):

            min-=1

        else:

            max+=1

    while (timeout < 0) or (timeout > 5):

        try:

            timeout=float(input(f"{bcolors.YELLOW}Time port (Default: 0.0025): {bcolors.GREEN}"))

        except KeyboardInterrupt:

            kill\_process()

        except:

            print(f"\n{bcolors.CYAN} Float number [0-5] (Default: 0.0025)\n Short time may cause the device to lag\n Big time will scan more accurately\n{bcolors.ENDC}")

            timeout=999

    while 1:

        try:

            type\_port=int(input(f"\n{bcolors.CYAN}Type scan:\n{bcolors.BLUE}1. TCP\n2. UDP\n{bcolors.YELLOW}Choose: {bcolors.GREEN}"))

            if (type\_port==1):

                type\_port="TCP"

                break

            elif (type\_port==2):

                type\_port="UDP"

                break

        except KeyboardInterrupt:

            kill\_process()

    main(host,min,max,timeout,type\_port)

*Ưu điểm:*

+ Dễ dàng sử dụng, tùy chỉnh

+ Tiện lợi, kiểm tra đơn giản đến phức tạp

+ Câu lệnh python thuần túy, tự chỉnh sửa như ip, quét tcp hay udp

*Nhược điểm:*

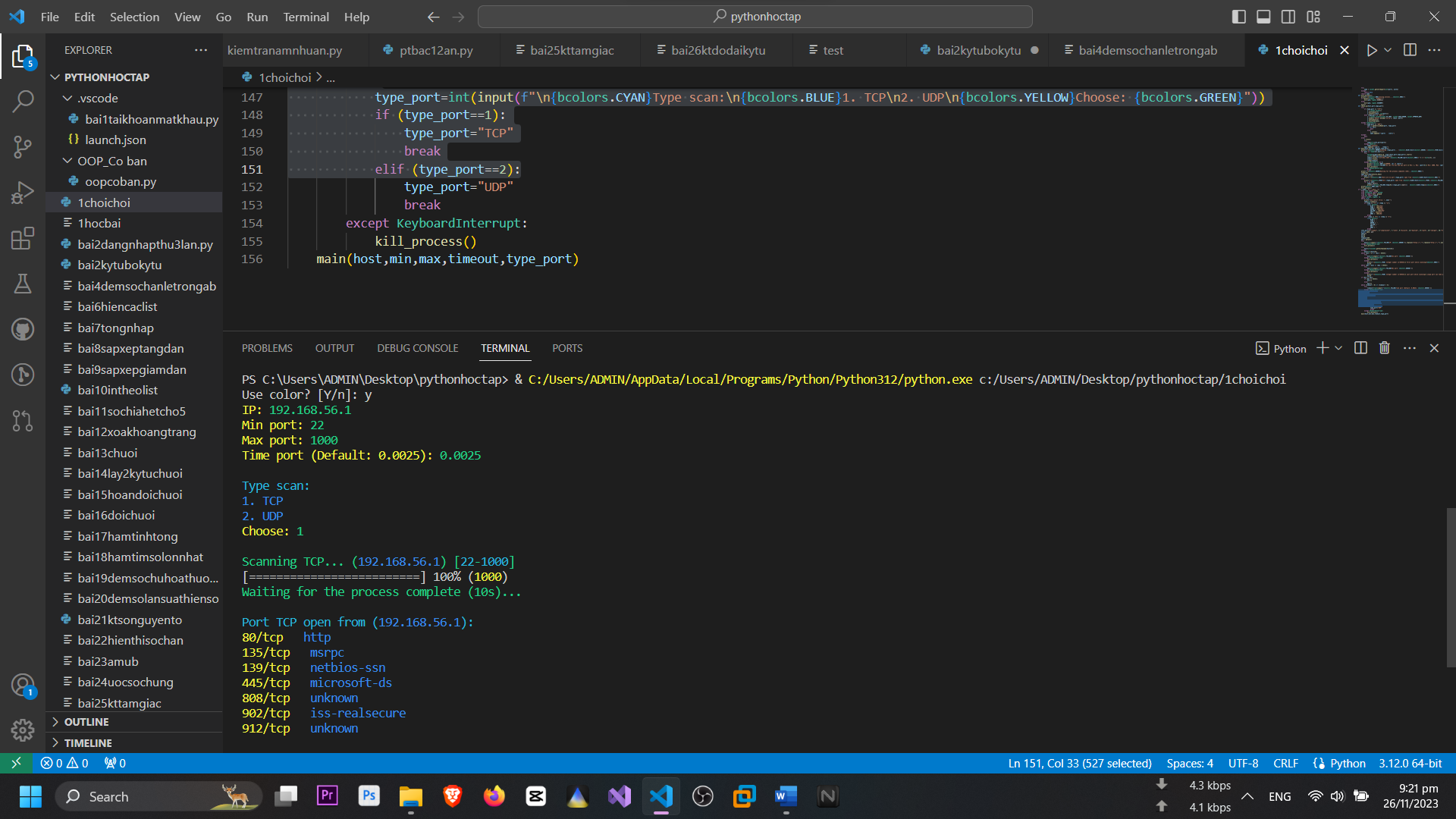
+ Một số cổng không biết chính xác -> unknown

+ Và chức năng quét không nhiều, ứng dụng chưa cao

Kết quả:



*Hình 2.1: Kết quả quét IP 127.0.0.1*



*Hình 2.2: Kết quả quét địa chỉ IP của máy*

**Kết luận**

*Kết quả đạt được*

Qua quá trình nghiên cứu, tìm hiểu, ứng dụng Python để xây dựng các tiện ích, các chương trình hỗ trợ để dò quét và khai thác phân tích cấu trúc của mạng máy tính:

- Quét và phân tích mạng cho phép sử dụng quét mạng để xác định các thiết bị kết nối, cổng mạng mà chúng đang sử dụng, và các dịch vụ đang chạy trên các cổng đó.

- Phát hiện thiết bị và máy chủ trong mạng xác định các máy chủ, thiết bị trong mạng bằng cách quét địa chỉ IP và phân tích thông tin thu thập được.

- Kiểm tra bảo mật hệ thống bằng cách kiểm tra các cổng mạng mở, các dịch vụ đang chạy trên hệ thống và kiểm tra xem chúng có bị lỗ hổng bảo mật không.

- Hỗ trợ phân tích xâm nhập trong quá trình phân tích xâm nhập để kiểm tra các điểm yếu và lỗ hổng bảo mật trên hệ thống.

Trong phạm vi khó có thể diễn đạt hết ý về mặt lý thuyết cũng như kỹ thuật, mặc dù đã cố gắng hoàn thành với tất cả sự nỗ lực của em, xong khó có thể tránh khỏi những thiếu sót. Kính mong nhận được những ý kiến đóng góp của thầy để em tiếp tục hoàn thiện kiến thức cũng như phương pháp của mình.

Em xin cam đoan những nội dung là sản phẩm của em tự nghiên cứu, những nội dung sao chép được chỉnh sửa theo ý hiểu. Nếu sai em xin hoàn toàn chịu trách nhiệm.

Em xin chân thành cảm ơn!